METHOD AND APPARATUS FOR ~~IDENTIFYING A PACKET~~ TRACING PACKETS

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION:

[001] The present invention relates generally to the field of network security and more specifically to using low overhead methods for identifying ~~packets~~the intrusion location of a packet in a network.

DESCRIPTION OF PRIOR ART:

[002] Availability of low cost computers, high speed networking products, and readily available network connections has helped fuel proliferation of the Internet. This proliferation has caused the Internet to become an essential tool for both the business community and private individuals. Dependence on the Internet arises, in part, because the Internet makes it possible for multitudes of users to access vast amounts of information and perform remote transactions expeditiously and efficiently. Along with [[the]]rapid growth of the Internet have come problems caused by malicious individuals or pranksters launching attacks from within the network. As the size of the Internet continues to grow, so does the threat posed by these individuals.

[003] The ever-increasing number of computers, routers and connections making up the Internet increases the number of vulnerability points from which these malicious individuals can launch attacks. These attacks can be focused on the Internet as a whole or on specific devices, such as hosts or computers, connected to the network. In fact, each router, switch, or computer connected to the Internet may be a potential entry point from which a malicious individual can launch an attack while remaining largely undetected. Attacks carried out on the Internet often consist of malicious packets being injected into the network. Malicious packets can be injected directly into the network by a computer, or a device attached to the network, such as a router or switch[[,]]. Such a

computer or device can be compromised and configured to place malicious packets onto the network.

[004] The most publicized forms of network attacks often involve placing thousands or millions of packets onto the network using a practice known as *flooding*. The flood of packets can be targeted to a specific device on the network, for example a corporate web site, thus causing the device to become overwhelmed and shutdown. Alternatively, an attack may be designed to clog the links, or connection points, between network components. Network attacks can be further enhanced using a practice known as *spoofing*. Spoofing involves associating bogus Internet Protocol (IP) addresses with [[the]]transmitted packets, thus making the packets' origins impossible to determine based upon looking only at a received packet. Spoofing can be further enhanced using a technique referred to as *transformation*. When a packet is transformed, it undergoes a process that changes the original packet into a new packet, as, for example, would happen during tunneling or network address translation (NAT). Locating the origin of a network attack is further complicated because *coordinated attacks* can be employed. In a coordinated attack, multiple network devices are compromised and then used to launch a *distributed attack*. A distributed attack is one that is launched essentially simultaneously from several locations within the network.

[005] Network attacks can also be launched using a single packet. While single packet attacks are not as well publicized as multi-packet attacks, they are becoming more common and they are capable of inflicting significant damage to vulnerable networks. At present, it is extremely difficult to detect single packet attacks in a timely manner using known methods of intrusion detection, which exacerbates the challenge in dealing with them. As a result, network data, currently, must be analyzed after the fact to determine if a single packet attack was the source of disruption. Any tracing of the single packet to its origins, in accordance with prior art techniques, must also take place after the attacking packet traversed the network.

[006] Much of the difficulty in identifying the origin of an attack arises because the Internet employs a stateless routing infrastructure, in that it is one in which routing is based solely on destination addresses. Although source IP addresses may be transmitted with data, they are easy to forge, and as a result they are untrustworthy. A forged source

address may bear no similarity to the actual source address from which the packet came. As a result, most prior art techniques and devices for preventing network attacks attempt to stop delivery of malicious packets at the ultimate destination device rather than attempting to locate their origin. Such origin is referred to as an *entry point*, also referred to as an *ingress point* or *intrusion location*, onto the network. Failing to identify the source address of malicious packets inhibits preventing further attacks, and such failure makes identification of the actual perpetrator difficult.

Figure 1

[007] Fig. 1 provides an example of a network employing prior art devices to thwart malicious packets. Two prior art autonomous systems are shown, PAS1 and PAS2, respectively, connected to the Internet, or public network (PN1) shown comprised of routers R2-R6. An autonomous system (AS) is a network domain in which all routers in the AS can exchange routing tables. Often the AS may be a local area network (LAN) such as one found at a university, municipality, large corporation, or Internet Service Provider (ISP). An AS may further be comprised of computers, or hosts, connected to the AS such as H1-H3 for PAS1 or H4-H5 for PAS2, respectively. An AS is normally connected to the public network by one or more border routers, here R1 (for PAS1) or a firewall F1 (for PAS2) incorporating router functionality.

[008] Border routers contain routing tables for other routers within the AS and for routers within the public network that are connected to the AS by a link, i.e. a communicative connection. In Fig. 1, R1 is a border router for PAS1 and it connects to the Internet using representative link L1. Routing tables act as road maps for routers on the network, in that they are used to ensure that network traffic is forwarded through the appropriate links in route to a desired destination address.

[009] Firewalls are typically installed between a local area network (LAN), or intranet, and the Internet, or public network. Firewalls act as gatekeepers for an AS in that they allow certain packets in while excluding other packets. Firewalls may be implemented in routers or servers connected between an AS and the Internet, or they may function as standalone devices. Rule sets are used by firewalls to determine which packets will be allowed into their respective AS and which packets will be discarded. Since rules determine which packets get through the firewalls, only packets known to be

problematic can be stopped. Therefore, rule sets must be updated on a regular basis to provide protection against new threat characteristics.

[0010] Additional protection for an AS may be obtained by supplementing border routers and firewalls with intrusion detection systems (IDSs). IDSs also use rule-based algorithms to determine if a given pattern of network traffic is abnormal. The general premise used by an IDS is that malicious network traffic will have a different pattern from normal, or legitimate, network traffic. Using a rule set, an IDS monitors inbound traffic to an AS. When a suspicious pattern or event is detected, the IDS may take remedial action, or it can instruct a border router or firewall to modify operation to address the malicious traffic pattern. For example, remedial actions may include disabling the link carrying malicious traffic, discarding packets coming from a particular source address, or discarding packets addressed to a particular destination. In Fig. 1, IDS1 is used to protect PAS1 and IDS2 is used in conjunction with F1 to protect PAS2.

[0011] Although border routers, firewalls, and IDSs can be used to help prevent known packets from entering an AS, they are not well equipped for stopping unknown packets because they rely on rule-based look up tables containing signatures of known threats. In addition, border routers, firewalls, and IDSs generally are not well equipped for identifying the origin, or ingress location, of malicious packets, particularly when spoofing is employed. Even when spoofing is not used, the above-noted devices may not be able to determine the ingress point for packets because packets often traverse many Internet links and devices, such as routers, bridges, and switches, before arriving at an AS. Reliably tracing the path of a packet often requires information about each link traversed by a packet. To obtain this information, routing data must remain with the packet or, alternatively, each router, or device, on the path must store information about, or a copy of, each packet traversing a network. With high-speed routers passing gigabits of data per second, storing full copies of packets is not practical.

[0012] What has been needed and what has not been available is a method for identifying the origin of malicious packets that can be implemented in an AS on the Internet and which addresses all shortcomings of prior art protection techniques. Embodiments of the present invention offer welcome solutions to these prior art protection problems.

SUMMARY OF THE INVENTION

[0013] Embodiments of the present invention employ apparatus, system, computer program product and/or method for identifying an intrusion point of a malicious or target packet into a network. ~~More specifically, in a network component operatively coupled to a network by at least one link carrying multiple packets, a computer-readable storage medium containing executable code for instructing a processor to process information about at least one of the packets, the information being used to facilitate locating an intrusion location for~~ More specifically, in a network including multiple hosts and multiple routers for facilitating transmission of packets on a network, a system, for example, is employed for determining the point of entry of a malicious packet. An intrusion detection system detects the entry of a malicious packet in the network. ~~A hash value is determined over at least a portion of one of the packets. The resulting hash value is used to form an index into a memory for storing information about a subset of the multiple packets. A flag is set at one of the memory locations corresponding to the index. A query containing information about a malicious packet is received and the information is extracted from the query. The information in the query is compared to the contents of the memory. And a reply is generated if the information in the query matches the contents of one of the memory locations, thus indicating that an intruding packet has been observed by the network component.~~ A source path isolation server responsive to the intrusion detection system isolates the malicious packet and thereby determines the point of entry of the malicious packet. In a further embodiment of the system, the source path isolation server includes a means for generating a query message containing information about the malicious packet and a means for forwarding the query message to some of the routers located one hop away. In still a further embodiment of the system, certain of the routers include means for generating a hash value of the identification information about the malicious packet, a means for establishing a bit map of hash values representative of packets having passed through the respective router, and a means for comparing the hash value of the identification information to the hash values of packets having passes through the respective router.

[001] ~~In a further aspect of the invention, in a network carrying multiple packets over at least one network link, where the network includes a computer, a first network component having memory and a processor configured to store information about at least one of the packets and a second network component, a target packet is detected. At least one of the multiple packets is received over a link to obtain a received packet. Next, a hash value or digest is determined over at least a portion of the received packet. The hash value is used to identify a memory location and a flag is set at the identified location. The first network component receives a query message identifying a target packet and uses the flag in processing the query message to determine if the target packet has been encountered. If the target packet has been encountered, the first network component replies.~~

[002] ~~In yet a further aspect of the invention, in a network carrying multiple packets over at least one link, where the network includes a network component having a memory and a processor, information about received packets is stored and at least a portion of the information is used to locate an intrusion point for one of the packets. A first packet is received at the first network component. A hash value is determined over at least a portion of the first packet. The hash value is used to identify a location in the memory and a flag is set at the location indicating that the hash value for the first packet has occurred. A second packet is received and processed to obtain information contained in it. This information is used to determine if the first packet has been observed. If the first packet has been observed, a reply is made available to the network and the reply may be used in a technique for locating the intrusion point for the first packet.~~

[003] ~~In still a further aspect of the invention, in a network carrying multiple packets over at least one link, the network including multiple devices and a system, the system being useful for assisting with the location of an intrusion point of a target packet in the network. The system having a first interface for receiving at least one of the multiple packets to produce a received packet. A second interface is used for placing a subset of any received packets onto the network link. A bus couples the first interface and second interface to allow communication. A memory is coupled to the bus, and the memory is used to store information about received packets in a machine-readable form. A processor is also coupled to the bus and the memory, and the processor is used to~~

~~execute machine-readable instructions for processing received packets. A first hash value is determined for each received packet. A second hash value is determined from at least a portion of a target packet. A first hash value is compared to the second hash value and a reply is made in response to the comparison.~~

[0014] In a further aspect of the invention, in a network carrying a plurality of packets where at least one of the packets is a target packet, the network includes at least one network component, a detection device and a server, a technique for determining the point of entry of a target packet into the network. The target packet is received from the detection device at the server. A query message is sent to a first one of the network components where the query message identifies the target packet. A reply containing information about the target packet from the first network component is received. The reply is processed to extract information contained therein. And, the information is used in a manner that allows the entry point of the target packet to ultimately be determined.

[0015] In yet a further aspect of the invention, in a network carrying a plurality of packets, a computer-readable data signal is embodied in a transmission medium used to identify an intrusion location of a target packet. The network includes a server and a network component having a memory storing representations of the plurality of packets, namely the data signal. A header portion includes an address of the network component. And, a body portion includes at least a portion of the target packet, the body portion being compared to corresponding representations where a match between a portion of the target packet and one of the representations indicates that the network component encountered the target packet.

[0016] In still a further aspect of the invention, in a network carrying a plurality of packets, the network includes a network component having a memory storing first information about a subset of the plurality of packets having passed through the network component. The network component further includes a processor for computing a first hash value of a target packet and a second hash value of a member of the subset of the plurality of packets. The memory also stores second information about an intrusion location of the target packet in the network. A data structure stored in the memory includes information resident in a database used by a source path isolation program for determining the intrusion location with the data structure. A network component

identification attribute corresponds to a location of the network component. A target packet attribute uniquely identifies the target packet. And, a reply packet attribute associated with at least one of the members and being associated with the network component identification attribute identifies the origin of the reply packet with the reply packet indicating that the member was encountered if the first hash value matches the second hash value.

[0017] It is advantageous to employ embodiments of the present invention to ~~eliminate~~protect data networks. A further advantage of the invention is the elimination of problems caused by undetected malicious packets in a network. A still further advantage of the invention is that it detects malicious packets without requiring special purpose network equipment. Furthermore, the present invention communicates information about malicious packets to other network devices thus enhancing network security. Another advantage of the invention is that ~~information about malicious packets is~~ it efficiently uses stored ~~thus facilitating cost effective deployment of disclosed embodiments.~~information about packets to facilitate detecting malicious packets.

[0018] It is thus [[an]]a general object of the present invention to provide improved packet networks.

[0019] It is another object of the present invention to eliminate [[the]]problems caused by malicious packets in a network.

[0020] It is a further object of the present invention to identify malicious packets to facilitate identifying their intrusion locations into the network.

[0021] It is [[yet]]a further object of the present invention to quickly identify [[the]]ingress points of malicious packets when distributed attacks are launched against a network.

[004] It is [[still]]yet a further object of the present invention to efficiently [[store]]use stored information about packets traversing a link in a network.

~~**[0022]** It is still yet a further object of the present invention to detect transformed packets.~~

[0023] Further objects and advantages of the ~~disclosed embodiments~~ present invention will become more apparent after reference to the detailed description of

exemplary embodiments thereof taken in conjunction with the accompanying drawings in which:

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] Fig. 1 is a block diagram of a prior art network comprising autonomous systems;

[0025] Fig. 2 is a block diagram of an exemplary embodiment of the present invention operating in conjunction with an Internet network;

[0026] Fig. 3 is a schematic diagram of an autonomous system coupled to a plurality of external networks;

[0027] Fig. 4 is a flowchart illustrating an exemplary method for use with a source path isolation server;

[005] Fig. 5 is a ~~flowchart illustrating an exemplary method for use with a source path isolation router;~~

[0028] ~~Fig. 6 is a~~ schematic diagram of an exemplary data structure for storing information ~~useable~~ in ~~conjunction with~~ a source path isolation server for use in performing source path isolation techniques; and

[0029] Fig. [[7]]6 is a block diagram of a general-purpose computer configurable for practicing exemplary embodiments or the invention.

DETAILED DESCRIPTION OF PREFERRED EMBODIMENT

Figure 2

[0030] A preferred embodiment uses a server and one or more specially configured network components, or devices, such as a router, within an autonomous system (AS) to determine the ingress point, or location, for a malicious packet (MP1).

Fig. 2 illustrates an embodiment that may be used with an Internet Protocol network. More particularly, Fig. 2 is broken into three general areas enclosed within borders. ~~The general areas within Fig. 2 are connected by links~~ with communication media, such as links, carrying data traffic across the network, connecting the general areas. Links[[can]] serve as a transmission media for data and signals on the network and may be comprised of wire, optical fiber ~~optic cable~~, radio frequency (RF) transponders, or the like.

[0031] The rightmost portion of Fig. 2 denotes an AS, shown as AS1, enhanced by the addition of a source path isolation server (SS1) and network components, here routers, modified to work as source path isolation routers (SRs), denoted by SR14-17, respectively. Also included within AS1 is a detection device, here an intrusion detection system (IDS) denoted as IDS1, and ~~destination components or devices such as~~ host computers H1-H3. IDS1 may take the form of a commercially available IDS, or alternatively it may be developed specifically for participating in source path isolation systems and methods. IDSs and firewalls are well known in the art and will not be described in detail herein. An informative source of information on IDS and firewall functionality that may be used with the disclosed embodiments can be found in *Firewalls and Internet Security: Repelling the Wily Hacker*, by William R. Cheswick and Steven M. Bellowin, Addison-Wesley (1994).

[0032] SS1 may be comprised of a ~~device such as a~~ general-purpose computer, or server, operatively coupled to the network of AS1 and executing machine-readable code enabling it to perform source path isolation in conjunction with SR14-17 and IDS1. While SS1 and IDS1 are shown as separate devices in Fig. 2, it is noted that they can be combined into a single unit performing both intrusion detection and source path isolation. SR14-17 may be comprised of commercially available routers, or similar devices such as switches, bridges or the like, employing software and hardware enabling them to participate in source path isolation. ~~Additional detail regarding the operation of an SR is described later herein.~~

[0033] The central portion of Fig. 2 represents the public network, shown as PN1, carrying [[data]]traffic between the autonomous systems, namely IAS1, and AS1, AS2 and AS3. PN1 comprises routers R2-R6, links ~~(shown as lines)~~ operatively coupling the routers making up PN1, and links attaching to ASs coupled to PN1. PN1 may also

comprise computers external to an AS (not shown). In the foregoing discussion, routers that have not been modified to act as source path isolation routers (SRs) are denoted as Rx, such as those located in PN1, where x is a number such as 2, 3, 4, etc..

[0034] The lower portion of Fig. 2 includes other autonomous systems, AS2 and AS3 that may be operatively connected to PN1. AS2 and AS3 may employ source path isolation apparatus and methods, or alternatively, they may be prior art autonomous systems (PAS).

[0035] The leftmost portion of Fig. 2 shows an autonomous system (IAS1) used by an intruder to launch an attack on AS1. IAS1 contains an IDS, shown as IDS2, operatively coupled to three host computers H4, H5 and I1 using links. In Fig. 2, I1 has been configured such that it places a malicious packet (MP1) onto IAS1 for transmission to AS1 via PN1. While Fig. 2 illustrates a ~~source, here a~~ computer[[,]] configured to place MP1 onto the network, routers, switches, gateways and other hardware capable of placing machine-readable data onto a network may be used ~~as sources.~~ in place of or in conjunction with such computer. When a device has been configured to inject ~~a malicious packet~~ an MP1 onto a network, it is referred to as an *intruder* or *intruding device*.

[0036] To launch an attack, an intruder generates malicious data traffic and places it onto a link for transmission to one or more destination devices having respective destination addresses. In Fig. 2, the heavy lines~~, denoting links,~~ are used to indicate the path taken by MP1, namely I1 to IDS2, IDS2-R6, R6-R3, R3-R2, R2-SR15, SR15-SR16, and SR16-IDS1 (where hyphenation implies operative coupling between network components). The thick dashed link from IDS1-H3 denotes the intended path to the ~~destination, or target,~~ targeted device H3.

[0037] Detection and source path isolation of MP1 may be accomplished as follows. Detection device, here IDS1, identifies MP1 using known methods. After detecting MP1, IDS1 ~~notifies~~ generates a notification packet, or triggering event, and sends it to SS1 thus notifying SS1 that a malicious packet has been detected within AS1. The notification packet may include MP1 or portions thereof along with other information useful for SS1 to begin source path isolation. Examples of information that may be sent from IDS1 to SS1 along with MP1 are time-of-arrival, encapsulation

information, link information, and the like. When MP1 (or fraction thereof) has been identified and forwarded to SS1 it is referred to as a target packet (TP1) because it becomes the target of the source path isolation method further described herein.

[0038] SS1 may then generate a query message (QM1) containing TP1, a portion thereof, or a representation of TP1 such as a hash value. After generating QM1 containing identification information about TP1, SS1 sends it to some, or all, participating routers. Accordingly, SS1 may send QM1 to participating routers located one hop away; however the disclosed invention is not limited to single hops. For example, SR16 is one hop away from SS1, whereas SR14, SR15 and SR17 are two hops away from SS1 and one hop away from SR16, respectively. When SR16 receives QM1 from SS1, SR16 determines if TP1 has been seen. This determination is made by ~~hashing TP1 and~~ comparing ~~the resulting hash value, or digest, to a bit map of hash values~~TP1 with a database containing signatures of other characteristics representative of packets having ~~previously~~ passed through SR16. Typically, SR16 is considered to have observed, or encountered, a packet when the packet is passed from one of[[the]]its input ports to one of [[the]]its output ports such as would be done when SR16 forwards~~, or propagates, a packet~~ during normal operation within a network.

[0039] To determine if a packet has been observed, SR16 first stores a representation of each packet it forwards. Then SR16 compares the stored representation to the information about TP1 contained in QM1. Typically, a representation of a packet passed through SR16 will not be a copy of the entire packet, but rather it will be comprised of a portion of the packet or some unique value representative of the packet. Since modern routers can pass gigabits of data per second, storing complete packets is not practical because memories ~~would have to be~~become prohibitively large. In contrast, storing a value representative of the contents of a packet uses memory in a [[much]]more efficient manner. By way of example, if incoming packets range in size from 256 bits to 1000 bits, a fixed width number may be computed across the bits making up a packet in a manner that allows the entire packet to be uniquely identified. A hash value, or hash digest, is an example of such a fixed width number. To further illustrate the use of representations, if a 32-bit hash ~~value, or~~ digest[[, may be]] is computed across each packet[[. Then,]], then the digest may be stored in memory or, alternatively, the digest

may be used as an index, or address, into memory. Using the digest, or an index derived therefrom, results in efficient use of memory while still allowing identification of each packet passing through a router ~~to be identified~~. The disclosed invention works with any storage scheme that saves information about each packet in a space efficient fashion, that can definitively determine if a packet has not been observed, and that will respond positively (i.e. in a predictable way) when a packet has been observed. Although the invention works with virtually any technique for deriving representations of packets, for brevity, the remaining discussion will use hash digests as exemplary representations of packets having passed through a participating router.

[0040] [[If]]Returning to the discussion of Fig. 2, if SR16 has not observed TP1, it may so inform SS1. But if SR16 has a hash matching TP1, it may send a response to SS1 indicating that the packet was observed by, or at, SR16. In addition, SR16 may forward QM1 to adjacent routers 1 hop away. In Fig. 2, SR16 sends QM1 to SR14, SR15 and SR17. Then, SR14, 15 and 17 determine if they have seen TP1 and notify SS1 accordingly. In this fashion, the query message/reply process is forwarded to virtually all SRs within an AS on a hop-by-hop basis.

[0041] In Fig. 2, routers SR14, SR15 and SR17 are border routers for AS1, namely they are the routers that contain routing tables for routers outside AS1. If routers external to AS1 have not been configured to operate as SRs, then the query message/reply process stops at SR14-17[[. But]]; however, if the public network routers are configured to act as SRs then the query message/reply process may continue until the SR closest to the ingress point of TP1 is reached. When the SR closest to the ingress point is found, it can be instructed to disconnect the link used by the intruder or it can be instructed to drop packets originating from the intruder's Internet Protocol (IP) address on a particular link, or based on other identifying information.

[0042] Still referring to Fig. 2 and the route taken by MP1, if the routers making up PN1 are not participating as [[SR's]]SRs, then SR15 would be instructed to exclude TPs. SR15 excludes a TP, present at an input port, by preventing it from passing to an output port. In contrast, if the routers making up PN1 were participating as SRs then R6 could be instructed to exclude TPs present at its input port.

[0043] The process used to perform source path isolation in Fig. 2 is referred to as an inward-out technique. After being triggered by an IDS, an inward-out technique begins its queries from a generally central portion of an AS. The inward-out technique then employs QMs that hop outward from the central portion of the AS toward the border routers comprised therein.

Figure 3

[0041] Fig. 3 illustrates an autonomous system (AS), 300, employing border routers denoted generally as B connected to external networks EN1-EN7, other routers within 300 connected to the border routers generally denoted as A, and a source path isolation server denoted as SS. AS 300 may also include additional routers (not shown) located between SS and border routers B. An inward-out solution begins with SS at the center of Fig. 3 and works outward one hop at a time until the border routers, B, are reached. For Fig. 3, the routers labeled A are queried on the first hop and the border routers, B, are queried on a second, or subsequent, hop.

[0044] Since the locations of border routers are known within AS 300, an outward-in solution may also be employed. With an outward-in solution, SS first queries the border routers, B, and they in turn query the routers labeled A. As can be seen from Fig. 3, an outward-in solution gets progressively closer to the center of AS 300. The disclosed technique [[may]]can be used on networks containing virtually any number of participating routers. While inward-out and outward-in techniques have been herein described, the disclosed techniques are not limited to any particular types of solution or localization algorithms. Furthermore, SS may send queries to participating routers located ~~more than one hop away enabling~~ virtually ~~any type~~anywhere in the network so that many types of source path isolation ~~technique to~~techniques can be employed. Thus it can be seen that the disclosed technique is very scalable and flexible.

[0045] Further detail of the operation of a source path isolation server (SS) and a source path isolation router (SR) are provided hereinbelow.

Figure 4

EXEMPLARY METHOD FOR SOURCE PATH ISOLATION SERVER

[0046] Fig. 4 illustrates an exemplary method for accomplishing source path isolation. The method begins when SS1 receives [[a]]TP1 from IDS1 operating within AS1 (step 402).

[0047] After receiving TP1, SS1 may generate QM1 comprising TP1 and any additional information desirable for facilitating communication with participating routers (SRs) (step 404). Examples of additional information that may be included in QM1 are, but are not limited to, destination addresses for participating routers, passwords required for querying a router, encryption keying information, time-to-live (TTL) fields, a hash digest of TP1, information for reconfiguring routers, and the like. SS1 may then send[[s]] QM1 to SRs located at least one hop away (step 406). SR may then process QM1 by hashing TP1 contained therein and comparing the resulting value to hash values stored in local memory, where the stored hash values identify packets having previously passed through SR.

[0048] After processing QM1, an SR may send a reply to SS1 (step 408). The response may indicate that a queried router has seen TP1, or alternatively, that it has not (step 410). It is important to observe that the two answers are not equal in their degree of certainty. If SR does not have a hash matching TP1, SR has definitively not seen TP1. However, if SR has a matching hash, then SR has seen TP1 or a packet that has the same hash as TP1. When two different packets, having different contents, hash to the same value it is referred to as a *hash collision*.

[0049] If a queried SR has seen TP1, a reply and identification (ID) information for the respective SR is associated as active path data (step 414). Alternatively, if an SR has not seen TP1, the reply is associated as inactive path data (step 412). Replies received from queried SRs are used to build a source path trace of ~~the potential~~ possible paths taken by TP1 ~~as it travels, or propagates, across a~~ through the network using known methods (step 416). ~~SS1 may attempt to build a trace with each received response~~ SS1 may then attempt to identify the ingress point for TP1 (step 418). If SS1 is unable to determine the ingress point [[for]]of TP1 ~~(step 418). If SS1 has not computed the ingress point, the~~ , subsequent responses from participating routers located an additional hop away are processed by executing steps 408-418 again (step 424).

[0050] Examples of source path tracing techniques that may be employed with embodiments disclosed herein are, but are not limited to, a breadth-first search or a depth-first search. In a breadth-first search, all SRs in an area are queried to determine which SRs may have observed a target packet. Then, one or more graphs, containing nodes, are generated from the responses received by SS1. Where the nodes indicate locations that TP1 may have passed. Any graphs containing a node where TP1 was observed are associated as active, or candidate, paths, i.e. paths that TP1 may have traversed. With a depth-first search, only SRs adjacent to a location where TP1 was observed are queried. SRs issuing a positive reply are treated as starting points for candidate graphs because they have observed TP1. Next, all SRs adjacent to those that responded with a positive reply are queried. The process of moving the query/response process out one hop at a time is referred to as a round. This process is repeated until all participating routers have been queried or all SRs in a round respond with a negative reply indicating that they have not observed TP1. When a negative reply is received, it is associated as inactive path data.

[0051] When SS1 has determined an ingress point for TP1, it may send a message to IDS1 indicating that a solution has been found (step 420). Often it will be desirable to have the participating router closest to the ingress point close off the ingress path used by TP1. As such, SS1 may send a message to the respective participating router instructing it to close off the ingress path using known techniques (step 422). SS1 may also archive ~~copies of~~ path solutions ~~generated~~, data sent, data received, and the like either locally or remotely. Furthermore, SS1 may communicate information about source path isolation attempts to devices at remote locations coupled to a network. For example, SS1 may communicate information to a network operations center (NOC), a redundant source path isolation server, or to a data analysis facility for post processing.

[0052] Here it is noted that as SS1 attempts to build a trace of the path taken by TP1, ~~several~~multiple paths may emerge as a result of hash collisions occurring in[[the]] participating routers. When [[hash]]collisions occur, they act as false positives in the sense that SS1 interprets the collision as an indication that a desired TP1 has been observed. Fortunately the occurrences of hash collisions can be mitigated. One mechanism for reducing hash collisions is to compute large hash values over the packets

since the chances of collisions rise as the number of bits comprising the hash value decreases. Another mechanism for reducing collisions is to control the density of the hash tables in the memories of participating routers. That is, rather than computing a single hash value and setting a single bit for an observed packet, a plurality of hash values are computed for each observed packet using several unique hash functions. This produces a corresponding number of unique hash values for each observed packet. While this approach fills the router's hash table at a faster rate, the reduction in the number of hash collisions makes the tradeoff worthwhile in many instances. ~~For example, Bloom Filters may be used to compute multiple hash values over a given packet in order to reduce the number of collisions and hence enhance the accuracy of traced paths. Therefore, the disclosed invention is not limited to any particular method of computing hash functions nor is it limited to a particular type of source path localization algorithm or technique.~~

~~EXEMPLARY SOURCE PATH ISOLATION ROUTER~~

~~**[0042]** To participate in source path isolation of target packets, a router is modified so that it can determine a hash value over the immutable portion of each packet received and/or forwarded. A router forwards a packet when it moves a data packet present at an input port to an output port for transmittal toward a desired destination. Modifying a router to record information about observed packets after computing a hash value provides an efficient method for retaining unique information about each packet seen, or observed, by a participating router. Techniques for quickly computing hash values are readily available and they can be implemented in the processing hardware and software currently used in routers without unduly reducing performance of the forwarding engines within the routers. In order to make use of hash value information, a participating router, SR, may store information in a manner facilitating rapid recall when QM1 is received from SS1. Since, modern routers are capable of forwarding large numbers of packets very quickly, attempting to store even a byte per data packet would require very large amounts of high-speed memory. Employing hash values significantly reduces the memory requirements for storing information about packets.~~

~~[0043] An SR determines a hash value over an immutable portion of a packet observed at an input port. The hash value is determined by taking an input block of data, such as a data packet, and processing it to obtain a numerical value that is unique for the given input data. The hash value, also referred to as a message digest or hash digest, is a fixed length whereas the input data may vary in size. Since the hash digest is unique for each input block of data, it serves as a signature for the data over which it was computed. For example, incoming packets varying in size from 32 bits to 1000 bits could have a fixed 32-bit hash value computed over their length. Furthermore, the hash value may be computed in such a way that it is a function of all of the bits making up the input data, or alternatively it can be computed over a portion of input data. When used, a hash value essentially acts as a fingerprint identifying the input block of data over which it was computed. However, unlike fingerprints, there is a chance that two very different pieces of data will hash to the same value, i.e. a hash collision. An acceptable hash function should provide a good distribution of values over a variety of data inputs in order to prevent these collisions. Since collisions occur when different, i.e. unique, input blocks result in the same hash value, an ambiguity arises when attempting to associate a result with a particular input. Suitable hash functions are readily known in the art and will not be discussed in detail herein. For example, hash functions used in the art, which may be used in conjunction with the matter disclosed herein, can be found in *Cryptography And Network Security: Principles And Practice*, Stallings, Prentice Hall (2000) and. An example of a useful hash function that can be used with the invention is the Cyclical Redundancy Check (CRC).~~

~~[0044] To further reduce collisions, each SR may implement its own unique hash function. By way of example, if there are two adjacent routers, SR15 and SR16, coupled together and each employs the same hash function, and there are two target packets, TP1 and TP2 on a network. Now assume, TP1 passes only through SR15, and TP2 passes through SR16 before arriving at SR15. If TP1 and TP2 have a hash collision at SR15, then the tracing algorithm will include SR16 in the traced path because SR16 would incorrectly report TP2's hash value as a potential signal that TP1 had passed through SR16 because of the identical hash values of TP1 and TP2. However, if SR16 employs a different hash function, then TP1 and TP2 will have different hash values at SR16, and~~

thus SR16 would not be included in the tracing path even though a collision occurred between TP1 and TP2 at SR15.

[0045] Generally packets have an immutable portion and a mutable portion. These names are used to help distinguish between the portions of the packet that may change as it is routed through the network and the portion, or portions, remaining intact, or unchanged. Immutable is used to describe the portions of a packet that do not change as a function of the packet's path across, or through, a network. In contrast, mutable describes the portions of a packet that change as a function of the packet's path through the network. Typically, the data, or baggage, portion of a packet is thought to be immutable whereas the header portion is considered to be mutable. Although the header portion may be largely comprised of mutable fields, it often contains immutable fields as well. When practicing the invention it is desirable to compute hash values over at least a portion of the immutable fields of a packet to produce hash values that do not change as the packet traverses a network.

[0046] Embodiments disclosed herein may store the actual hash values to identify packets traversing the network, or they may use other techniques for minimizing storage requirements associated with retaining hash values and other information associated therewith. One such technique for minimizing storage requirements uses a bit array for storing hash values. Rather than storing the actual hash value, which can typically be on the order of 32 bits or more in length, the invention uses the hash value as an index for addressing into a bit array. In other words, when a hash value is computed for a forwarded packet, the hash value serves as the address location into the bit array. At the address corresponding to the hash value, a single bit is set at the respective location thus indicating that a particular hash value, and hence a particular data packet, has been seen by the router. For example, using a 32 bit hash value provides on the order of 4.3 billion possible index values into a bit array. Storing one bit per packet rather than storing the packet itself, which can be 1000 bits long, produces a storage ratio of 1:1000. While bit arrays are described by way of example, it will be obvious to those skilled in the relevant art, that other storage techniques may be employed with out departing from the spirit of the invention.

[0047] While using a bit array significantly reduces memory requirements for participating routers, they are not eliminated. Over time, a memory will fill up and the possibility of overwriting an existing index value increases. The risk of overwriting an index value may be reduced if the bit array is periodically flushed to other storage media such as a magnetic disk drive, optical media, solid state drive, or the like. To facilitate this, a time-table may be established for flushing the bit array, wherein such time-table may be based on the speed of the router, number of input data streams, the size of available fast memory, and the like. If desired, the flushing cycle can be reduced by computing hash values only for a subset of the packets passing through a router. While this approach reduces the flushing cycle, it increases the possibility that a target packet may be missed, i.e. a hash value is not computed over a portion of it.

Figure 5

[0048] Fig. 5 presents an exemplary method for an SR. While the foregoing discussion utilizes a single SR, namely SR1, it will be readily apparent to those skilled in the art that multiple SR's may be queried, and hence perform source path isolation essentially simultaneously. SR1 receives QM1 from SS1 within AS1 (step 502). After receiving QM1, SR1 may determine if a time-to-live (TTL) field in the query is expired (step 504). If a TTL field is used, QM1 is discarded if the TTL field is expired (step 506). A TTL field is normally employed because it provides an efficient mechanism for ensuring that SR1 responds only to relevant, or timely, queries. In addition, employing TTL fields reduces the amount of data traversing the network between SS1 and participating routers.

[0049] If the TTL field is not expired, SR1 determines if TP1 has been transformed (step 508). TP1 is transformed when it undergoes a transformation in route through a network such that a hash value computed over the immutable portion of the packet has a different value from that of the non-transformed portion. For example, TP1 may have undergone a transformation of the baggage portion of the packet in an attempt to make identification of TP1 and/or its source more difficult. If TP1 has been transformed, SR1 creates a new query packet (QM2) containing a hash value for the immutable portion of the transformed packet (step 510). Where no packet transformation has occurred, the method determines if the hash value computed matches an index value

~~in the bit array (step 512). As previously noted, index values contained in the bit array identify hash values of packets that have been forwarded by a queried router, here SR1. Depending on available memory in SR1, the hash value may be compared to bit array indices retrieved either from disk or from volatile memory.~~

~~[0050] If the hash value does not match an index value, SR1 does not forward QM1 (step 516), but instead may send a negative reply to SS1 (step 518). If a queried SR determines that TP1 has been transformed, the hash value of this variant, referred to as QM2, may be added to the baggage portion of QM1 (step 514), or alternatively can be used to create a new message (not shown) for forwarding to other devices. Next, QM1 is preferably forwarded to all interfaces excluding the one that QM1 was received on (step 520). After forwarding the message, SR1 sends a positive reply to SS1 indicating that the packet has been observed (step 522). The reply may contain the address of SR1, information about observed packets, and information about transformed packets, such as QM2, that have passed through SR1.~~

~~Figure 6~~

EXEMPLARY DATA STRUCTURE FOR STORING [[PACKET]]TRACE INFORMATION

~~[0051]~~ Fig. [[6]]5 illustrates an exemplary data ~~structures for storing information about TPs that have passed through SR1. Data structures can be divided into records, shown as R(1) - R(n) to facilitate storage and retrieval. Additionally, data structures may be~~ structure 500 stored in ~~volatile or non-volatile~~ a database (not shown) in a memory~~. Since fast, or volatile, memory is expensive, it may be desirable to store incoming data to fast memory for a specified period of time that is a function of the memory size and incoming data rate. Often the time period will be chosen such that it ends when~~ on a ~~selected record size is filled. When the record is full, it may be closed and flushed to less costly disk storage. Upon flushing a record from memory, a new record is opened and information about incoming data is stored. The flushing and opening of records can be overlapped in time to ensure that all incoming packets are accounted for. To further aid in efficient recall of flushed records, it may be desirable to time stamp records when they~~

~~are opened. Time stamping records provides a convenient mechanism for tracking them in both volatile and non-volatile memory. In Fig. 6, R(1) is being filled in a fast memory such as RAM, and records R(2) - R(n) are stored to disk or other storage device.~~

~~[0052] As previously disclosed herein, a hash value is preferably determined over an immutable portion of TP1 when it passes through SR1, and the resulting hash value is used as an index value, or address, into a memory. The index value is used to facilitate the storage of information about a packet so that it can be uniquely identified. In Fig. 6, the index values are represented by E(0) - E(n) where n is the last entry location in memory. Since the hash value is used as the address, the number of available addresses will be determined by the size of the hash value used, for example 32, 64 or 128 bits. It is desirable that the memory be sized so that likelihood of an index being overwritten is very small. Also, it is beneficial to set all of the data storage locations to the same initial value, say "0", before storing data into memory, where "0" indicates that a packet has not yet been observed.~~

~~[0053] When a hash value is determined for a particular TP1, an indicator bit, or flag, is set at an address corresponding to that hash value. The indicator bit is used to confirm that a particular TP1 has either been "seen" or "not-seen". If a hash value is computed for a TP1, then the indicator bit is set to some state, for example to a "1". The "1" indicates that the respective TP1 has been "seen" by SR1. In Fig. 6, an indicator bit entry with a "1" denotes that the corresponding hash value has been seen, while a "0" indicates that a corresponding hash value has not been seen.~~

[0053] ~~Additional information may be stored for each hash value, or for a given data table or record, to further aid with~~ source path isolation ~~for TP1. For example, a "time" parameter can be associated with each computed hash value. If used, "time" will normally represent the exact time that a particular TP1 was seen by SR1. Additionally, a "link id" parameter can~~server. Data structure 500 stores information used in conjunction with performing source path isolation of a target packet. While Fig. 5 illustrates one data structure, it will be obvious to those skilled in the relevant arts that a plurality of data structures may be employed and that the data structures may include additional parameters and take on different forms from those of the exemplary data structure discussed herein.

[0054] Data structure 500 is comprised of a record R(1) containing attributes, or parameters, having data associated therewith. In the upper left portion of Fig. 5 are three parameters associated with the entire record R(1), namely a target packet attribute, shown as Target ID, a time attribute, shown as Time, and a source attribute, shown as Source. These attributes together serve as a handle for R(1) to facilitate storage into, and recall from, a machine-readable memory (not shown). Here Target ID is associated with unique information associated with a particular target packet (TP) received from a detection device such as an IDS or firewall. Time may be used to identify either the time at which TP was received at an SS, the time that TP was received at a detection device, or the time that R(1) was opened. Source may be used to identify the link that TP was detected on by the detection device, or alternatively, source may be used to uniquely identify the detection device that forwarded TP to SS.

[0055] Within 500 are exemplary column headings indicating still other attributes that may be used to facilitate source path isolation of TP. For example, a network component identification attribute, shown as node ID, may be used to identify particular nodes, such as routers, switches, bridges, or the like, within a network that have been queried by SS. Link may be used to identify the particular link ~~upon which a TP1 arrived. Identifying the link may be of benefit when SR1 disables an ingress path for TP1. A "status" parameter can be employed to aid with monitoring system performance and health. It will be apparent to those skilled in the art that numerous other parameters can be associated with data arriving at SR without departing from the spirit of the invention~~on which TP was observed. A reply packet attribute, shown as Node Response, may be used to indicate if a queried node has observed TP. Node time may indicate the time, preferably using some common reference, at which a respective node observed TP. Time is useful for assessing how long TP has been in the network and for performing comparisons with fields such as time-to-live (TTL). The attribute Transformed is used to track variants of TP in the event it has undergone a transformation. If TP has been transformed, it may be useful to have multiple entries associated the respective TP. For example in Fig. 5, node 04 has two entries for tracing an untransformed and a transformed version of TP. Status may be used to monitor network links associated with queried nodes. For example, a status of "ON" may indicate that a link is still active, i.e.

carrying data traffic, while a status of "OFF" may indicate that a link has been disabled to exclude data traffic.

[0056] Fig. 5 illustrates one exemplary embodiment of a data structure that may be used for facilitating source path isolation; however, variations of the data structure format and number of records may be readily employed without departing from the spirit of the invention. For example, the terms "YES/NO" and "ON/OFF" used in conjunction with node response, transformed, and status may be desirable when conveying information to an operator; however, flags such as 1 or 0 may also be used to indicate the status of various attributes. In addition, a plurality of records may be generated when performing source path isolation. Additionally, other column entries may be used in conjunction with, or in place of, those shown in Fig. 5. For example, it may be desirable to associate the hash value, or alternatively, the contents of TP with each record. It may also be desirable to have a record associated with each target packet encountered or, alternatively, with each detection device employed within a network. And, it may be desirable to have still other data structures or records associated with source path solutions that have been generated in response to detected TPs.

Figure [[7]]6

EXEMPLARY SYSTEM FOR PERFORMING METHOD

[0057] FIG. [[7]]6 illustrates a system [[720]]620 comprising a general-purpose computer that can be configured to practice disclosed embodiments. System [[720]]620 executes machine-readable code to perform the methods heretofore disclosed and[[it]] includes a processor [[702]]602, main memory [[704]]604, read only memory (ROM) [[706]]606, storage device [[708]]608, bus [[710]]610, display [[712]]612, keyboard [[714]]614, cursor control [[716]]616, and communication interface [[718]]618.

[0058] Processor [[702]]602 may be any type of conventional processing device that interprets and executes instructions. Main memory [[704]]604 may be a random access memory (RAM) or a similar dynamic storage device. Main memory [[704]]604 stores information and instructions to be executed by processor [[702.]]602. Main memory [[704]]604 may also be used for storing temporary variables or other intermediate information during execution of instructions by processor [[702.]]602.

ROM [[706]]606 stores static information and instructions for processor [[702.]]602. It will be appreciated that ROM [[706]]606 may be replaced with some other type of static storage device. Storage device [[708]]608, also referred to as data storage device, may include any type of magnetic or optical media and their corresponding interfaces and operational hardware. Storage device [[708]]608 stores information and instructions for use by processor [[702.]]602. Bus [[710]]610 includes a set of hardware lines (conductors, optical fibers, or the like) that allow for data transfer among the components of system [[720.]]620.

[0059] Display device [[712]]612 may be a cathode ray tube (CRT), liquid crystal display (LCD) or the like, for displaying information in an operator or machine-readable form. ~~The keyboard 714~~Keyboard 614 and cursor control [[716]]616 allow the operator to interact with system ~~720. The cursor~~620. Cursor control [[716]]616 may be, for example, a mouse. In an alternative configuration, keyboard [[714]]614 and cursor control [[716]]616 can be replaced with a microphone and voice recognition means to enable an operator or machine to interact with system [[720.]]620.

[0060] Communication interface [[718]]618 enables system [[720]]620 to communicate with other devices/systems via any communications medium. For example, communication interface [[718]]618 may be a modem, an Ethernet interface to a LAN, an interface to the Internet, a printer interface, etc.. Alternatively, communication interface [[718]]618 can be any other interface that enables communication between system [[720]]620 and other devices, systems or networks. Communication interface [[718]]618 can be used in lieu of keyboard [[714]]614 and cursor control [[716]]616 to facilitate operator or machine remote control and communication with system [[720]]620.

[0054] As will be described in detail below, system [[720]]620 may provide SS1 operating within AS1 with the ability to perform source path isolation for a given TP[[1]]. SS1 may receive MP1 from IDS1 and generate QM1 in response to processor [[702]]602 executing sequences of instructions contained in, for example, memory [[704.]]604. Such instructions may be read into memory [[704]]604 from another computer-readable medium, such as storage device [[708]]608, or from another device coupled to bus 710610 or coupled via communication interface [[718.]]618. Execution of sequences of instructions contained in memory [[704]]604 causes processor [[702]]602 to

perform the method described in conjunction with FIG. 4. For example, processor [[702]]602 may execute instructions to perform the functions of ~~determining~~ receiving a ~~hash value for MP1~~ target packet (step 402), receiving replies from queried routers (step 408), and building a trace of the path traveled by TP[[1]] (step 416). Alternatively, hard-wired circuitry may be used in place of or in combination with software instructions to implement the functions of SS[[.]]1. Thus, the disclosed embodiments of [[SS]]SS1 are not limited to any specific combination of hardware circuitry and software.

[0061] ~~System 720 may also be used to enable SR1 to pass data, store information about packets that have been seen, and respond to query messages. For example, SR1 may compute, or determine, a hash value over an immutable portion of a packet using processor 702 and instructions received from, for example, memory 704. The execution of instructions contained in memory 704 causes processor 702 to further perform the method generally described in Fig. 5, such as receiving a QM1 (step 502), evaluating the time-to-live field of QM1 (step 504), determining if a packet has been transformed (step 508) and sending a reply to SS1 (step 522). Hard-wired circuitry can also be used in place of or in combination with software instructions to implement the functions of SR1. Thus the disclosed embodiments of SR1 are not limited to any specific combination of hardware and software.~~ For example, the functionality may be implemented in an application specific integrated circuit (ASIC), a field-programmable gate array (FPGA), or the like, either alone or in combination with other devices to provide desired functionality.

CONCLUSION

[0062] As can be seen, the disclosed embodiments provide the functionality necessary to facilitate [[the]]source path isolation of malicious packets in a network. While the preceding disclosure is directed to an Internet Protocol (IP) network, disclosed embodiments can be used in conjunction with other network protocols such as frame relay, asynchronous transfer mode (ATM), synchronous optical network (SONET), and the like. In addition, disclosed embodiments may be adapted to operate within different layers of a network such as the data link layer, network layer, transport layer or the like.

Furthermore, the disclosed embodiments are not limited to particular network topologies or architectures.

~~[0055] Also, methods associated with determining a hash value for packets that have been seen may be implemented in various types of network devices in addition to source path isolation routers (SRs) heretofore discussed. For example, the method discussed in conjunction with Fig. 5 may be implemented in network switches, bridges, hubs, or within monitoring equipment coupled to a network. In addition,~~ Furthermore the disclosed methods ~~can be implemented in dedicated hardware subsystems or processors to provide enhancements to legacy equipment. Furthermore, the disclosed methods can operate on encapsulated data such as would be encountered if network data were encrypted, converted from one network protocol to another, or a packet was split for transmission over more than one link.~~

[0063] ~~The disclosed methods~~ for implementing a source path isolation server (SS) ~~and a source path isolation router (SR)~~ are not limited to a single programming language or hardware architecture. For example, software for performing the functions of SS [[or SR]]may be implemented in a high level programming language such as C, C++, LISP, or the like. Alternatively, software may be implemented in a lower level language such as assembly language, or a device specific language, where requirements such as speed must be met. Furthermore, [[an]]SS [[or SR]]may be configured to communicate with, and make information available to, other devices operatively connected to a network using known programming languages and techniques. For example, it may be desirable to have SS make source path isolation solutions available to an operator responsible for monitoring network security. In addition, [[an]]SS [[or SR]]can be implemented in a distributed fashion either by employing multiple processors or by having various components physically separated and coupled by a communication means such as a distributed bus, network, or the like. Also, it may be desirable to have [[an]]SS communicate with one or more SRs over a dedicated network instead of using the network carrying data traffic among the SRs. For example, using a dedicated network may provide additional security, reliable bandwidth, or communication redundancy in the event that one or more links to an SR is disabled.

[0064] Query messages (QMs) and replies are not limited to a single network protocol or packet type. In many instances, it will be desirable to have QMs and replies transported using readily known protocols; however, customized protocols and message types can be used. For example, it may be desirable to employ a *smart packet* for sending QMs to participating routers. A smart packet is one that may contain a standard message, such as the data from a target packet, along with machine-readable ~~code containing executable~~ instructions for instructing a receiving device, such as an SR, to modify its operation in response to the contents of the executable [[code]]instructions contained ~~within the smart packet,~~therein. Smart packets facilitate rapid responses to network intrusions by allowing an SR to modify operation soon after receiving a QM from [[an]]as SS, or a forwarded QM from a participating router.

[0065] Furthermore, the disclosed methods can operate on encapsulated data such as would be encountered if network data were encrypted, converted from one network protocol to another, or a packet was split for transmission over more than one link. As can be seen, many variations of the disclosed embodiments are possible without departing from the spirit of the invention.

[0066] Therefore, the present embodiments are to be considered in all respects as illustrative and not restrictive, the scope of the invention being indicated by the appended claims rather than by the foregoing description, and all changes within the meaning and range of equivalency of the claims are therefore intended to be embraced therein.